[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/50

[12] 发明专利申请公开说明书

[21] 申请号 200410000575.X

[43] 公开日 2004 年 8 月 4 日

[11] 公开号 CN 1517872A

[22] 申请日 2004.1.14
[21] 申请号 200410000575.X
[30] 优先权
[32] 2003.1.16 [33] US [31] 10/346，147
[71] 申请人 国际商业机器公司
地址 美国纽约
[72] 发明人 P·J·海尔曼 K·R·赫普勒
H·J·梅 K·C·沃森

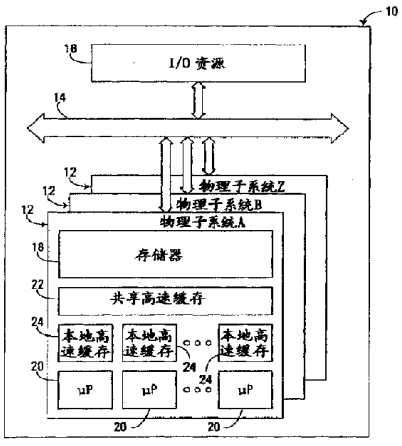
[74] 专利代理机构 北京市中咨律师事务所
代理人 于 静 李 峥

权利要求书 4 页 说明书 13 页 附图 3 页

[54] 发明名称 动态分配计算机资源的方法和装置

[57] 摘要

一种装置、程序代码和方法，根据与线程关联的特定“类型”把这些线程动态地指定到包括多个物理子系统的多线程计算机中的计算机资源。具体地说，按线程类型分配驻留在计算机的同一物理子系统内的资源，从而那些特定线程类型的新建线程和/或再激活线程动态地被指定到分配给它们各自线程类型的资源。这样，共享同一类型的那些线程通常被指定到驻留在计算机的同一物理子系统内的计算机资源，这往往减少驻留在一个计算机中的多个物理子系统之间的交叉通信量，从而改善系统总体性能。



ISSN1008-4274

1. 一种在包括多个不同物理子系统的多线程计算机中动态分配计算机资源的方法，该方法包括：

（a）对于多个线程类型当中的每个线程类型，使物理上位于该多线程计算机中一个共同的物理子系统内的一组计算机资源与这一线程类型相关联；以及

（b）响应激活一个线程的请求，把该线程指定到与该线程的线程类型相关联的那组计算机资源。

2. 权利要求1的方法，其中当还没有任何计算机资源与这多个线程类型中的第一个线程类型关联时，使这多个线程类型中的第一个线程类型与该组计算机资源关联是响应对一个具有该第一个线程类型的线程的激活请求而进行的。

3. 权利要求1的方法，其中使这多个线程类型中的第一个线程类型与该组计算机资源关联是在接收对一个具有这第一个线程类型的线程的激活请求之前进行的。

4. 权利要求3的方法，其中使这多个线程类型中的第一个线程类型与该组计算机资源关联是响应为这第一个线程类型预先分配资源的请求而进行的。

5. 权利要求3的方法，其中使这多个线程类型中的第一个线程类型与该组计算机资源关联是响应在该多线程计算机中启动一个逻辑子系统的请求而进行的。

6. 权利要求1的方法，其中多个线程类型是根据从这样一组特征中选出的一特征来区分的，这组特征包括运行优先级、对缓冲区的指定、用户标识、用户简档、存储器子系统、父任务、父线程、父作业、父应用、父逻辑子系统、用户授权以及它们的组合。

7. 权利要求1的方法，其中每个物理子系统包括一个存储器

和至少一个处理器。

8. 权利要求 7 的方法，其中每个物理子系统包括多个处理器，其中至少一部分存储器由这多个处理器共享。

9. 权利要求 7 的方法，其中每个物理子系统还包括一个高速缓存。

10. 权利要求 7 的方法，其中每个物理子系统包括一个唯一的多芯片模块（MCM）。

11. 权利要求 1 的方法，其中所述多个线程类型中的第一个线程类型与物理上位于所述多个物理子系统当中的第一个物理子系统内的第一组计算机资源关联，该方法进一步包括：

（a）使物理上位于该多线程计算机中不同于所述第一个物理子系统的一个物理子系统内的第二组计算机资源与所述第一个线程类型关联；以及

（b）响应激活该第一个线程类型的一个线程的请求，把这个线程指定到与该第一个线程类型关联的所述第一组和所述第二组计算机资源之一。

12. 一种在包括多个不同物理子系统的多线程计算机中动态分配计算机资源的方法，该方法包括：

（a）对于多个线程类型当中的第一个线程类型，使物理上分别位于该多线程计算机中第一个和第二个物理子系统内的第一组和第二组计算机资源与这第一个线程类型相关联；

（b）响应激活该第一个线程类型的第一个线程的请求，把这第一个线程指定到所述第一组计算机资源，使得在该第一个线程执行过程中该第一个线程利用该第一组计算机资源中的资源；以及

（c）响应激活该第一个线程类型的第二个线程的请求，把这第二个线程指定到所述第二组计算机资源，使得在该第二个线程执行过程中该第二个线程利用该第二组计算机资源中的资源。

13. 一种装置，包括：

（a）一个多线程计算机，其包括多个不同的物理子系统；以及

（b）驻留在该多线程计算机上的程序代码，其被配置成：对于多个线程类型当中的每个线程类型，使物理上位于该多线程计算机中的一个共同物理子系统内的一组计算机资源与这一线程类型相关联；以及响应激活一个线程的请求，把该线程指定到与该线程的线程类型相关联的那组计算机资源。

14. 权利要求 13 的装置，其中该程序代码被配置成：当还没有任何计算机资源与这多个线程类型中的第一个线程类型关联时，响应对一个具有该第一个线程类型的线程的激活请求使这多个线程类型中的该第一个线程类型与该组计算机资源相关联。

15. 权利要求 13 的装置，其中该程序代码被配置成：在接收对一个具有这多个线程类型中第一个线程类型的线程的激活请求之前，使该第一个线程类型与该组计算机资源相关联。

16. 权利要求 15 的装置，其中该程序代码被配置成：响应为这多个线程类型中第一个线程类型预先分配资源的请求，使该第一个线程类型与该组计算机资源相关联。

17. 权利要求 15 的装置，其中该程序代码被配置成：响应在该多线程计算机中启动一个逻辑子系统的请求，使这多个线程类型中的第一个线程类型与该组计算机资源相关联。

18. 权利要求 13 的装置，其中多个线程类型是根据从这样一组特征中选出的一特征来区分的，该组特征包括运行优先级、对缓冲区的指定、用户标识、用户简档、存储器子系统、父任务、父线程、父作业、父应用、父逻辑子系统、用户授权以及它们的组合。

19. 权利要求 13 的装置，其中每个物理子系统包括一个存储器和至少一个处理器。

20. 权利要求 19 的装置，其中每个物理子系统包括多个处理器，其中至少一部分存储器由这多个处理器共享。

21. 权利要求 19 的装置，其中每个物理子系统还包括一个高速缓存。

22. 权利要求 19 的装置，其中每个物理子系统包括一个唯一的多芯片模块（MCM）。

23. 权利要求 13 的装置，其中所述多个线程类型中的第一个线程类型与物理上位于所述多个物理子系统当中的第一个物理子系统内的第一组计算机资源关联，该程序代码进一步被配置成：使物理上位于该多线程计算机中不同于所述第一个物理子系统的一个物理子系统内的第二组计算机资源与该第一个线程类型关联；以及响应激活该第一个线程类型的一个线程的请求，把这个线程指定到与该第一个线程类型关联的该第一组和该第二组计算机资源之一。

24. 一个程序产品，其包含：

（a）驻留在包括多个不同物理子系统的一类多线程计算机上的程序代码，该程序代码被配置成：对于多个线程类型当中的每个线程类型，使物理上位于该多线程计算机中一个共同物理子系统内的一组计算机资源与这一线程类型相关联；以及响应激活一个线程的请求，把该线程指定到与该线程的线程类型相关联的那组计算机资源；以及

（b）承载该程序代码的信号承载介质。

25. 权利要求 24 的程序产品，其中所述信号承载介质包括传输介质和可记录介质二者当中的至少一个。

动态分配计算机资源的方法和装置

技术领域

本发明涉及计算机和计算机软件，特别是涉及多线程计算机中的计算机资源分配。

背景技术

由于当代社会中对计算机的依赖不断增长，计算机技术不得不在许多前沿取得进步，以跟上不断增长的需求。大量研究与开发工作的一个特别课题是并行化，即多个任务并行执行的性能。

已开发出若干计算机软件和硬件技术，以有利于增强并行处理。从软件的角度看，已开发出多线程操作系统和内核，它们允许计算机程序以多“线程”并发执行，从而使多个任务基本上能同时进行。线程通常代表程序的一些独立的执行路径。例如，对于一个电子商务计算机应用，不同的线程可被指定到不同的客户，从而每个客户的指定电子商务事务可以在一单独的线程中得到处理。

从硬件角度看，计算机越来越依赖于多个微处理器，以提供增强的工作负荷能力。再有，已开发出某些微处理器，它们支持并行执行多个线程的能力，在效果上提供了通过使用多个微处理器所能得到的许多同样的性能增益。

然而，在多处理器计算机中出现的一个重要的瓶颈与进、出每个微处理器的数据传送相关联，这个瓶颈通常被称作通信成本。大多数计算机依靠主存储器，其作为该计算机的主要工作存储器。然而，从主存储器中检索数据和把数据存回主存储器往往被要求以明显低于微处理器内部数据传送速度的速度进行。当微处理器使用数据时，往往利用称作高速缓存的中

间缓冲器来暂时存储来自主存储器的数据。这些高速缓存往往比主存储器尺寸小但速度显著较快。高速缓存往往利用数据的时间和空间的局部性，结果往往显著减少计算机中发生的比较慢的对主存储器的访问次数并减少了计算机承受的总通信成本。

在一个计算机中的全部微处理器往往共享同一个主存储器，这种体系结构通常称作对称多处理（SMP）。然而，这类计算机的一个限制是由于通常要求在一个公共总线或互连上进行多个微处理器与主存储器之间的所有通信。随着计算机中微处理器个数的增加，到主存储器的通信量成为计算机性能的瓶颈，其与是否使用中间高速缓存无关。

为克服这一潜在的瓶颈，许多计算机设计依靠非均衡存储器访问（NUMA），以此使多个主存储器基本上分布在一个计算机上并与各组微处理器和高速缓存组合成物理子系统或模块。一个 NUMA 计算机的每个物理子系统中的微处理器、高速缓存和存储器通常被安装在同一电路板或电路卡上，以提供物理子系统内的所有“本地”部件之间较高速的交互。这些物理子系统还通过网络（如一系统总线或一些点到点互连的集合）彼此连接，从而允许一个物理子系统中的微处理器访问存储在另一物理子系统中的数据，这样便能有效地扩展计算机的总体能力。然而，由于对存储在本地存储器（即与微处理器处在同一物理子系统中的存储器）中的数据的访问时间往往显著地短于对存储在远程存储器（即处在另一物理子系统中的存储器）中的数据的访问时间，所以这种存储器访问被称作“非均衡的”。

所以，从通信成本的角度看，在 NUMA 系统中，通过使数据通信本地化于每个物理子系统内并最大限度地减少需在各物理子系统之间传送数据的次数，使 NUMA 系统中的性能达到最佳。

有效地利用计算机中的硬件资源往往需要软件和硬件之间的协调努力。如前所指，从软件的角度看，由计算机进行的大部分工作由各线程处理。为保证最佳性能，线程通常被指定到各可用计算机资源子集，其指定方式是要使计算机的工作负荷在各可用计算机资源上均匀分布。

例如，为了有效地利用微处理器，希望在可用微处理器当中均匀分布线程，以平衡每一单个微处理器的工作负荷，这一过程称作“对称”资源分配。然而，由于通信成本也能对系统性能造成显著影响，所以也希望逻辑上把一个线程与它将使用的数据绑在一起，于是，只要可能，便把线程对数据的访问本地化在一个高速缓存中，或者，如果是在一个 NUMA 计算机中，则至少本地化在同一物理子系统之内。否则，访问非本地化数据的通信成本可能超过线程对称分布所带来的好处。通常，数据与线程的捆绑需要由人决定把通用型线程与物理上本地化的存储器、处理器以及相关资源关联起来。

在对称资源管理方案中，线程是在激活时，例如每当创建或再激活线程的时候，被分配的。被激活的线程通常被指定到最可能得到的或负荷最小的资源或资源组。然而，为解决通信成本问题进行的资源（如存储器资源）非均衡分布通常不是以这种自动和透明的方式实现的。而是，非均衡资源管理往往需要大量的用户分析和定制配置，包括例如计算机程序的定制编程以具体解决资源分配问题。

资源管理更希望在计算机的操作系统级或内核级进行，从而不依赖于用到可能安装在计算机上的应用或其他计算机程序上的任何特定编程技术。具体地说，当把资源管理嵌入操作系统或内核中时，这样的资源管理将不需要在较高级计算机程序中进行特定的定制以支持计算机资源的最佳分配，从而对可能在给定计算机上执行的所有计算机程序提供性能上的好处。特别是在 NUMA 计算机中，那里通过把线程利用的资源本地化在单个物理子系统中而获得性能上的好处，将更加希望以更透明的方式实现有效的资源分配，而无需显著的定制。

发明内容

本发明通过提供一种装置、程序产品和方法来解决与现有技术有关的这些及其他问题，在这一装置、程序产品和方法中，根据与线程关联的特定“类型”把线程动态地指定到各计算机资源。具体地说，按线程类型分

配驻留在计算机的同一物理子系统内的资源，从而那些特定类型的新建线程和/或再激活线程被动态地指定到分配给它们各自线程类型的资源。这样，共享同一类型的那些线程通常被指定到驻留在计算机的同一物理子系统内的计算机资源，而且往往是以基本上透明的方式，而无需对这些线程所关联的计算机程序进行任何显著的定制。

如在下文中将清楚看到的那样，可以利用很多线程属性或特征把线程分类成各种线程类型。除了其他可能的区分外，可根据运行优先级、对缓冲区的指定、用户标识、用户简档、存储器子系统、父任务、父线程、父作业、父应用以及用户授权等属性中的一个或多个来定义线程类型。

在许多实例中，这里描述的线程指定将显著减少一特定物理子系统内的资源与驻留在其他物理子系统中的其他资源交互的需求。而是，一个特定线程的通信量和处理开销更可能局限在单个物理子系统内，从而消除伴随子系统之间通信的开销并使系统性能达到最好。

在一些实施例中，与给定线程类型相关联的资源可被限定在单一物理子系统内。然而，在其他一些实施例中，一个线程类型可被分配来自多个物理子系统的资源。然而，在后一些实施例中，可能仍然希望把具有给定线程类型的各个线程指定到驻留在已为该线程类型分配了资源的那些物理子系统中的只一个子系统上的那些资源。

在这里所附的并构成本说明书又一部分的权利要求中列举了表征本发明的这些和其他特点和优点。然而，为了更好地理解本发明以及通过使用本发明所获得的好处和达到的目的，应参考附图以及相关的描述，其中描述了本发明的实施示例。

附图说明

图 1 是一多线程计算机中主要硬件组件的方框图，在该计算机中纳入了根据本发明的动态资源分配。

图 2 是图 1 的多线程计算机中主要软件组件的方框图。

图 3 是流程图，显示由图 2 中引用的线程调度程序执行的激活线程例

程的程序流。

图 4 是流程图，显示由图 2 中引用的线程调度程序执行的另一个激活线程例程的程序流。

图 5 是流程图，显示由操作系统执行的启动应用/逻辑子系统例程的程序流，该例程与图 4 所示激活线程例程结合使用。

图 6A-6D 是方框图，显示按本发明的方式在一多线程计算机的两个物理子系统之间分配计算机资源的示例。

图 7 是方框图，显示按本发明的方式在一多线程计算机的两个物理子系统之间分配计算机资源的另一示例，其显示把来自不只一个物理子系统的计算机资源分配给一个特定的线程类型。

具体实施方式

下文讨论的实施例利用了一种动态资源分配机制，这种机制至少是部分地根据线程类型在包括多个物理子系统的那类多线程计算机中把资源分配给那些线程。根据本发明，每当一个特定线程类型与特定一组资源关联时，所有其后被激活并与那个线程类型匹配的线程将被指定到这同一组资源。再有，这些资源组通常被限定于单个物理子系统，以便最大限度地减少计算机的多个物理子系统之间的交叉通信量，从而使总体系统性能最优化。

这样，在根据本发明的实施例中，通常只有当创建唯一类型的线程时，才会发生资源对称指定；否则，对于和先前已被分配资源的其他线程共享相同线程类型的那些线程，发生的是非对称指定。与其他非均衡资源指定相似，按类型对线程分组通常将提供一个好处，即增大容易地得到一特定线程所需资源而无显著延时的可能性。

在下文讨论的实施例中，通常能在一计算机系统中比较透明地实现资源指定。一旦定义了线程分类规则，通常往往能在无需明显的用户管理的情况下进行资源指定。这与传统的 NUMA 体系结构相反，NUMA 体系结构往往需要明显的配置、定制和用户分析。这也与传统的对称资源管理方

案相反，其中线程通常被指定到当这些线程被创建或重新激活时最可用的那些资源，而不管多个线程之间有任何特定的相似性。

下文的讨论将把被指定到特定资源组的实体称作“线程”。然而，应该理解，其他术语可用于描述那些定义一计算机系统中唯一执行路径的实体。这样，术语“线程”应被认为对应于计算机中的定义计算机系统中特定执行路径的任何实体。

在本发明的上下文中，线程类型实际上可包括一个线程的任何属性或其他区别特征，其包括但不限于运行优先级、对同一虚拟或物理缓冲区或存储器的指定、用户标识、父逻辑子系统、父作业、父应用、父任务或父线程、对同一存储器子系统的指定、线程启动时要执行的初始程序的名称、线程优先级以及用户简档。

再有，一组资源可包括处理器、本地高速缓存、共享高速缓存、共享存储器等计算机资源的任何组合。此外，计算机资源可包括其他形式的资源，如各种输入/输出（I/O）资源。通常，适于通过非对称相似性作为组指定的资源位于一个计算机的特定的与其他物理子系统不同的一个物理子系统内，这里一个子系统通常被认为是一组计算机资源，它们彼此交互比和其他物理子系统中的资源交互更有效。例如，在下文讨论的实施例中，物理子系统由位于同一模块中的硬件资源组合来确定，例如位于同一电路卡或多芯片模块（MCM）上，或由这同一电路卡或多芯片模块直接访问或以其它方式控制。例如，在来自国际商业机器公司的 eServer iSeries 中型计算机系统中，物理子系统可包括含有多个处理器和一个共享存储器以及各级（例如 L1、L2 和/或 L3）中间共享的和/或本地的高速缓存的唯一的或与其他不同的 MCM。再有，在一些实例中，存储器可放置在与一物理子系统的其余组件分离的卡上，然而可使用驻留在 MCM 上的控制器直接访问。在这样的实施例中，一个给定 MCM 上的资源之间的通信往往比不同 MCM 上的资源之间的通信快得多。这样，把归为同类的线程指定到放置在单一 MCM 上的资源可使 MCM 之间的通信减至最少而有利于增加的 MCM 内部通信，从而使计算机系统的总体性能最佳化。

在所示实施例中，每当线程被激活，例如每当它们被创建（如果是新线程）和/或每当它们被重新激活（如果已存在，但当前处于未激活或休眠状态）时，它们便被指定到特定的资源组。然而，在不同的实施例中，可在不同的时间点进行对特定线程类型分配资源组。例如，在一个实施例中，可随着尚未分配资源和/或需要添加资源的那种类型的一个线程的激活，为一个线程类型指定资源。然而，在其他实施例中，可在激活一种类型的任何线程之前为那种线程类型指定资源，例如，随着一个应用的启动、一个逻辑子系统的启动或响应为特定线程类型预分配资源的一特定的程序指令。在这方面，逻辑子系统实际上可包括逻辑上彼此相关的应用、作业、线程或任务的任何集合。

再有，如在下文中将更清楚看到的那样，在一些实施例中，一个线程类型可被分配一组与一计算机系统中的多个物理子系统关联的计算机资源。例如，当一个特定线程类型预计需要比在一个给定物理子系统中可得到的资源还多的资源时，可能希望分配来自多个物理子系统的计算机资源。还有，当放置在一个物理子系统中的资源明显不够用时，可能希望允许多个物理子系统分担负荷。然而，即使在这类情况中，可能希望定义子类型，或者至少是把一给定类型的特定线程分配给只位于一个物理子系统上的资源组（例如，对于一个特定线程，把该线程指定到位于一单个 MCM 上的诸处理器和存储器）。

在又一些实施例中，线程类型可指定到这样一组计算机资源，该组资源只包括一个给定物理子系统中可得到的计算机资源的子集。

现在转到附图，其中若干图中的相似数字代表相似组件，图 1 显示一个计算机 10 中的主要硬件组件，在该计算机 10 中纳入了根据本发明的动态资源分配。计算机 10 一般性地代表若干诸如网络服务器、中型计算机、大型计算机等的多用户计算机中的任何一个，如 AS/4000 或 eServer iSeries 中型计算机。然而，应该理解，本发明可在其他计算机或数据处理系统中实现，如在诸如工作站、桌面计算机、便携计算机等的单用户计算机中，或在其他可编程电子设备（如包含嵌入式控制器等）中实现。

计算机 10 一般包括多个物理子系统 12，它们通过系统总线或其他通信接口彼此连接。此外，计算机 10 通常包括各种 I/O 资源，其一般化地以 16 表示，包括诸如存储设备、工作站、终端、网络、成像设备等各类资源。

每个物理子系统 12 包括一组计算机资源，如共享存储器 18、一个或多个微处理器 20 以及一级或多级高速缓冲存储器，如共享高速缓存 22 以及一个或多个本地高速缓存 24。每个物理子系统 12 中的资源 18-24 的特点是它们彼此之间进行的交互或通信与它们和其他物理子系统 12 中的资源进行的交互或通信相比更有效。例如，在每个物理子系统 12 中的资源可放置在同一个多芯片模块（MCM）或电路卡上，从而使这些资源之间的互连可以比连接于系统总线 14 的互连快一个数量级或更快。

在本发明的其他实施例中可利用计算机资源的其他物理分区。再有，其他计算机硬件体系结构也可利用这里讨论的动态资源分配技术。所以，本发明不限于图 1 中所示的具体硬件实现。

接下来，图 2 显示能在图 1 的计算机 10 中被利用的软件体系结构示例 30。如图中所示，体系结构 30 可依靠操作系统 32，在该操作系统上执行多个作业或应用。在一些实施例中，在一个共同的逻辑子系统 36 中的一个或多个应用可被彼此关联，而另一些应用可能不与任何特定的逻辑子系统关联。

如图 2 中所示，作为多线程计算机的计算机 10 可执行或处理多个线程 38，以代表用户去完成所请求的任务。线程 38 可在若干情境内被利用，包括：在操作系统 32 内、在一特定应用 34 内、在一特定逻辑子系统 36 内和/或在体系结构 30 中的其他地方。在一些体系结构中，甚至可以定义多个逻辑分区，从而在一个给定体系结构中可发现多个包括独立执行的操作系统的逻辑分区。所以，实际上线程 38 可逻辑上驻留在给定计算机体系结构中的任何地方。

对线程 38 的管理，包括向特定线程分配资源，通常由线程调度程序 40 完成，其通常驻留在操作系统 32 内。例如，在上文讨论的 eServer iSeries 实现中，线程调度程序 40 可驻留在这样的计算机中的特许内部代码（LIC）

中。还可以理解，在根据本发明的一些实施例中，一个线程调度程序可能只管理一个计算机中可能线程的一个子集。

一般地，为实现本发明的实施例而执行的例程，不论是实现为一个操作系统的一部分还是实现为一个具体应用、组件、程序、对象、模块或指令序列，或者甚至是它们的子集，在这里都被称作“计算机程序代码”，或简单地称作“程序代码”。程序代码通常包含一个或多个指令，它们在各个时间驻留在一个计算机中的各个存储器和存储设备中，而且当由计算机中的一个或多个处理器读取和执行时，这些指令使计算机进行各种必要的步骤，以执行体现本发明各方面的步骤或单元。再有，尽管本发明已经和将要在全功能计算机和计算机系统的情境内被描述，但本领域技术人员将会理解，本发明的各种实施例能作为程序产品以多种形式分发，而且不管实际用于进行这种分发的信号承载介质的特定类型是什么，本发明都同样适用。信号承载介质的实例包括但不限于易失性和非易失性存储设备、软盘和其他可卸盘、硬盘驱动盘、磁带、光盘（例如 CD-ROM、DVD 等）等可记录型介质以及数字和模拟通信链路等传输型介质。

再有，下文描述的各种程序代码可根据在本发明的特定实施例中该程序代码在其中实现的应用或软件组件加以标识。然而，应该理解，所遵循的任何特定的程序命名都只是为了方便而使用的，因此本发明不应被局限于只是应用于由这种命名所标识和/或隐含的任何特定应用。再有，由于通常可有无穷多的方式把计算机程序组织成例程、过程、方法、模块、对象等，而且有各种方式在典型计算机内驻留的各种软件层（操作系统、库、API 应用、小应用程序等）当中分配程序功能，所以应该理解，本发明不限于这里描述的程序功能的特定组织和分配方式。

本领域技术人员将会理解，图 1 和图 2 所示环境示例不是要限制本发明。相反，本领域技术人员将会理解，可以使用其他替代硬件和/或软件环境而不脱离本发明的范围。

现在转向本发明的具体实施例，图 3 显示一个激活线程例程示例 50，该例程可以由图 2 的线程调度程序 40 响应于一激活（例如创建或重新激活）

线程请求而执行。例程 50 在块 52 开始，在那里确定与该线程关联的线程类型。如前文指出的那样，很多线程特征和属性可用于按类型区分线程。例如，下文讨论的例子集中在由父应用或逻辑子系统定义的线程类型，于是为一特定应用或逻辑子系统而启动的所有线程将共用相同的类型。一般而言，对于线程的任何特征或属性，如果据此而来的一类线程被分配共同的资源组而不是不同的资源组，会产生较好的性能，则该特征或属性可被用于对线程进行根据本发明的分类。

一旦确定了线程类型，接下来块 54 确定所确定的线程类型是否已被分配了一组资源。首先，假定尚未有资源分配给那个确定的线程类型，则块 54 将把控制交给块 56，以把一个物理子系统上的一组资源指定给所确定的线程类型，例如通过对称指定被本地化到单个或已知一组物理子系统的最可得到的一组资源。例如，当不同的资源组与不同的物理子系统关联时，块 56 可把一最可得到的物理子系统的诸资源指定给所确定的线程类型。再有，如果分配单一物理子系统上的资源是不可能的或不现实的，则该线程类型能被透明地分成多个子类型，以使一个类型与多个物理子系统上的不同组资源关联。

一旦资源已被分配给该线程类型，接下来块 58 把要激活的线程指定到已经为所确定的线程类型分配的那些资源。然后该线程被以传统方式激活，于是例程 50 完成。

回到块 54，如果已经为所确定的线程类型分配了资源，则块 56 可被越过，控制可直接传给块 58，以把这新线程指定到先前为那个线程类型分配的资源。在另一种作法中，如图 3 所示，块 54 可把控制传给块 60，以动态确定该线程类型是否需要附加资源。例如，如果与一特定线程类型关联的资源已被完全利用，则可能希望把控制传给块 56，以便向该线程分配附加资源（通常，如果可能，则使用放置在同一物理子系统中的资源）。否则，控制可从块 60 传到块 58。然而，在另一些实施例中，可能不支持附加资源的动态分配。

使用图 3 中的例程 50，可以看到对一线程类型的资源分配与激活这样

一个线程相联系，该线程具有的类型先前尚未被分配资源。然而，作为图3的线程激活例程的替代作法，可能希望在与线程激活分开的一个单独操作中对给定线程类型预先分配计算机资源。例如，在eServer iSeries计算机中，当用户作业被启动时，通常这些用户作业要被指定到一个物理子系统内的存储器池。然而，当逻辑子系统被初始化或被启动时，存储器池通常被指定到这些逻辑子系统。所以，在一个计算机中可以得到多组存储器和处理器的一个实施例中（例如当诸物理子系统被定义在多个MCM上时），可能希望在一个逻辑子系统被初始化或被启动时，只要可能，便试图把一特定存储器池的全部存储器分配在单个MCM或物理子系统上。再有，如果一个存储器池太大，则可定义诸子存储器池并使其位于各单个MCM上，但通常是以对用户透明的方式进行。可以预见，根据本发明，一个线程调度程序可被配置成使在一个关联的子系统或存储器池中运行的作业高度亲合那些与该子系统或存储器池相关联的处理器。

例如，图4和图5显示另一个实施例，其中向线程类型分配资源是随着一个父应用或逻辑子系统的启动进行的。如图4中所示，在这一实施例中的激活线程例程70可通过在块72中确定线程类型来执行，这类似于图3的块52中进行的过程。然而，一旦线程类型被确定，便认为资源已分配给所确定的类型。这样，控制可直接传到块74以把这新的线程指定到已分配给那个线程类型的资源。然后例程70便完成了。还应该理解，在这一不同的实施例中，如果希望的话，也可检验是否需要附加的资源。

如图5中所示，每当收到一个启动应用或逻辑子系统的请求时，便可执行启动应用/逻辑子系统例程80。例程80在块82开始，在那里创建一个与正启动的该应用或逻辑子系统关联的线程类型。接下来，块84为这新创建的线程类型分配一个物理子系统上的资源，例如使用对称指定，而且通常使用位于一单个物理子系统中的资源。如前文指出的那样，如果分配单一物理子系统上的资源是不可能的或不现实的，则该线程类型能被透明地分成多个子类型，以使一个类型与多个物理子系统上的不同组资源关联。

接下来，块86启动所请求的应用或逻辑子系统，激活所希望的任何线

程，于是例程 80 完成。可以理解，对于在块 86 中激活的任何线程，通常例程 70 将被调用，以便激活那个线程，如上文中结合图 4 描述的那样。

下面结合图 6A-6D 描述实现本发明的一种方式的实例。假定计算机有两个物理子系统，例如处理器卡或 MCM，其中每个包含多个处理器和一定容量的存储器。每个物理子系统的存储器可由所有处理器访问，但当从与发请求的处理器相同的物理子系统获取信息时，对存储器的访问最快。结果，如果每个处理器的绝大多数存储器访问都能被本地化，那会是有好处的。在这种情况下，资源组是具有相关联的处理器、高速缓存以及存储器的卡。在图 6A 中，这两个物理子系统示为 PS A 和 PS B，每个物理子系统的资源组由处理器（P）、高速缓存（C）以及存储器（M）资源表示。

现在转到图 6B，假定一个逻辑子系统被启动以支持第一个应收帐款应用，其将有 50 个线程运行。因为根据本发明全部 50 个线程被“分类”为相似的，这些线程可全部被指定到第一个物理子系统 PS A。如图 6B 中加到 PS A 中的资源上的阴影所代表的那样，通常在该物理子系统中的所有处理器和高速缓存都可由这个应收帐款应用中的诸线程使用。再有，通常一个代表该物理子系统中可用存储器子集的存储器池也可供这些线程使用。

然而，应该理解，在一些实施例中，在这个应收帐款应用中的线程可能不总是只被指定到第一个物理子系统。特别是，应该理解，可能存在这样一些情况，使得希望把一个线程或应用所利用的资源扩展到它被指定到的资源之外，例如，如果这个应收帐款应用是在该系统上执行的唯一应用的话。线程和它们的资源之间的“亲合”概念往往要求指定规则不总是一成不变的，而是可以随时间根据具体场合的要求而改变。

现在转到图 6C，假定在另一个逻辑子系统中启动了第二个应用，这是一个存货控制应用。为了这一例子的目的，在这一应用中的线程的“类型”被认为不同于与应收帐款应用相关联的线程的类型（由于它们驻留在一单独的逻辑子系统这一情况）。这样，可使用对称指定把这些线程指定到最少使用的一组资源（在这一情况中是第二个物理子系统 PS B），造成对处

理器和高速缓存资源以及存储器池的分配，如图 6C 中的附加阴影所示。

接下来，转到图 6D，假定第三个应用被启动，这是一个顾客信息应用，与这一应用的类型关联的线程将被指定到任何最少活动的处理器/存储器组。为了这个例子的目的，假定第二个物理子系统是最少活动的。如图 6D 中的附加阴影所示，处理器和高速缓存资源被分配给与存货控制和顾客信息两个应用亲合的线程类型使用。然而，通常为每个线程类型保持驻留在同一物理子系统中的单独的存储器池。

应该指出，尽管在全局的意义上资源可能没有被均匀地使用，但单个线程将趋向于即使在利用程度较高的资源上也能更有效地操作，因为它们将能有效地访问正由它们的线程“类型”处理的数据。在一天结束时，应收帐款应用的活动显著减少，而工资单应用被启动，工资单“类型”将透明地、自动地指定到最可得到的资源，这可能是先前被应收帐款应用频繁地使用过的资源。然而，当需要大量额外的应收帐款工作，因而在启动工资单应用之前该应用的活动没有下降时，自动指定到可能选择其他处理器/存储器资源组，而无需任何操作员或系统管理员干预。

接下来，图 7 显示一个与结合图 6A-6D 讨论的例子相似的例子。然而，在图 7 的例子中，第一个线程类型被分配来自两个物理子系统的资源，而第二个线程类型被分配只来自一个物理子系统的资源。例如，可能是这样一种情况：一个应收帐款应用需要的资源多于单个物理子系统所能提供的资源，而存货控制应用有较低的资源需求。将会理解，尽管应收帐款线程类型被分配来自多个物理子系统的资源，但那个类型的各单个线程将被透明地分成“子类型”，这些子类型将被指定到只来自一个物理子系统的资源，从而趋向于在逐个线程的基础上维持资源本地性。可根据指定给同一基础类型的各个物理子系统的总体可用性，把该基础类型的新线程指定到一个特定的子类型。这一可用性可基于不同的因子，包括例如线程计数、处理器利用程度和/或存储器利用程度。

利用所公开的内容，对本领域技术人员而言，其他修改将是显然的。所以，本发明取决于所附权利要求。

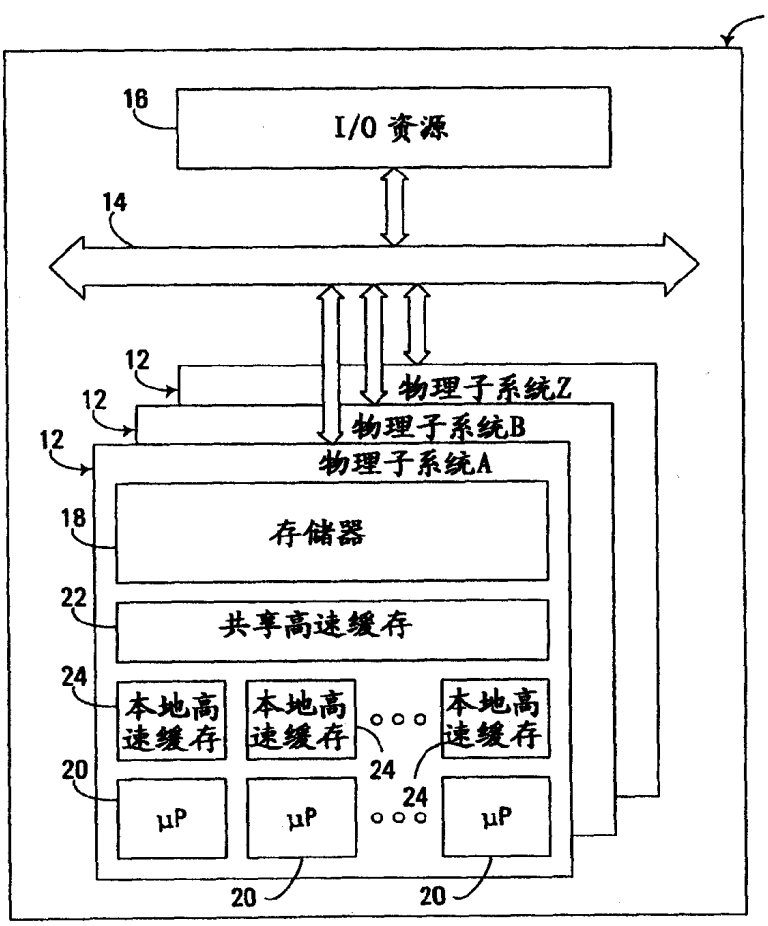

10
16
I/O 资源
14
12
物理子系统Z
12
物理子系统B
12
物理子系统A
18
存储器
22
共享高速缓存
24
本地高速缓存
本地高速缓存
本地高速缓存
24
24
20
μP
μP
μP
20
20

图1

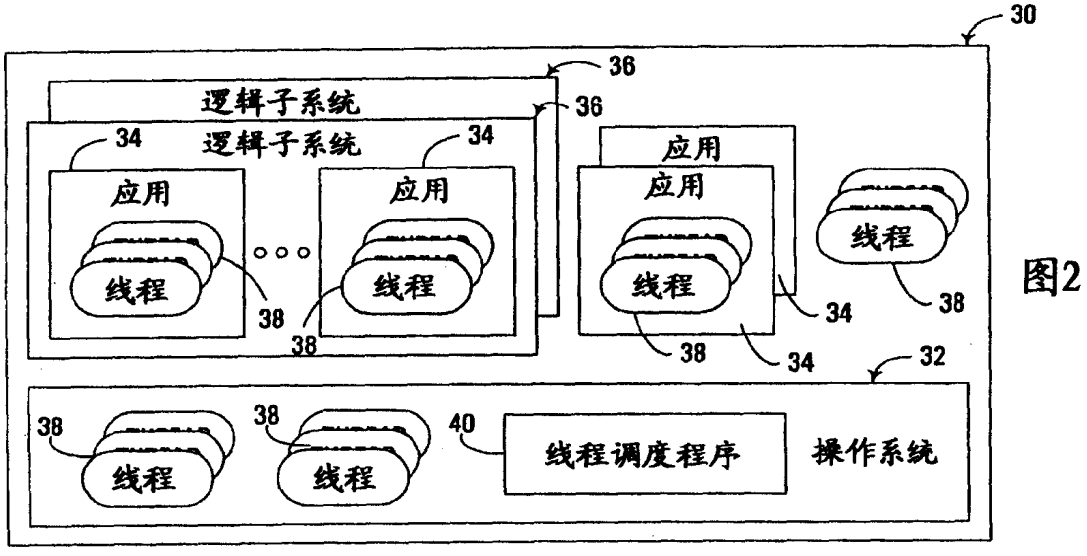

30
36
逻辑子系统
36
逻辑子系统
34
34
应用
应用
应用
应用
线程
线程
线程
线程
38
38
38
34
34
38
32
38
38
40
线程
线程
线程调度程序
操作系统

图2

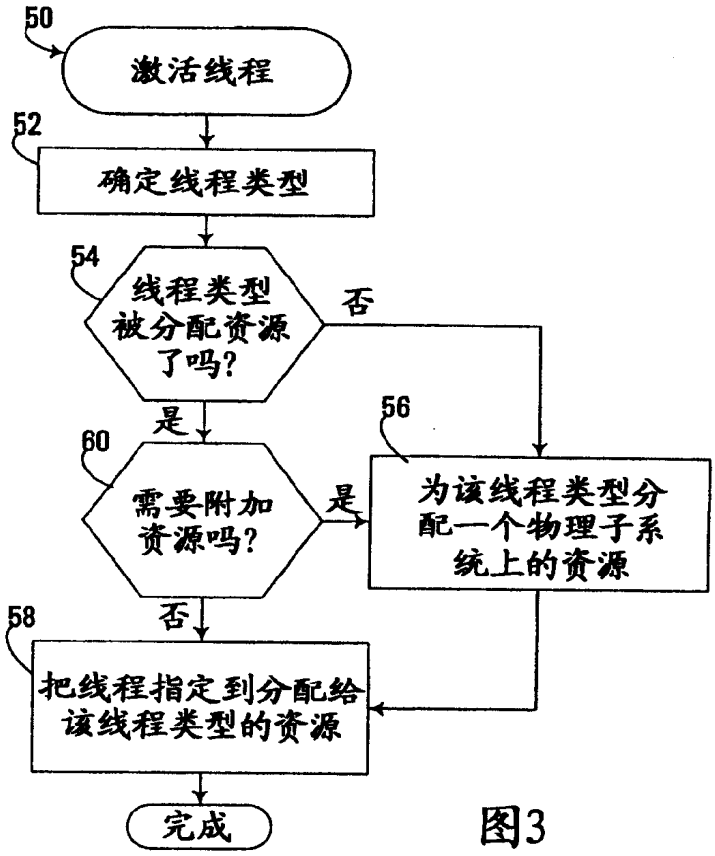

50
激活线程
52
确定线程类型
54
线程类型被分配资源了吗?
否
是
56
为该线程类型分配一个物理子系统上的资源
60
需要附加资源吗?
是
否
58
把线程指定到分配给该线程类型的资源
完成

图3

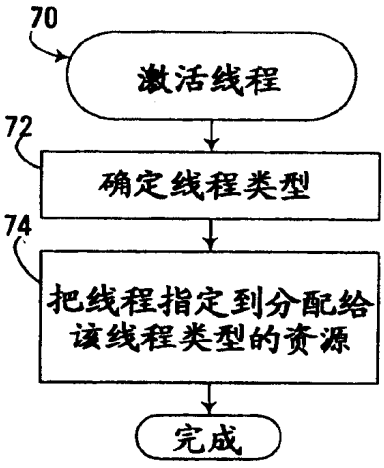

70
激活线程
72
确定线程类型
74
把线程指定到分配给该线程类型的资源
完成

图4

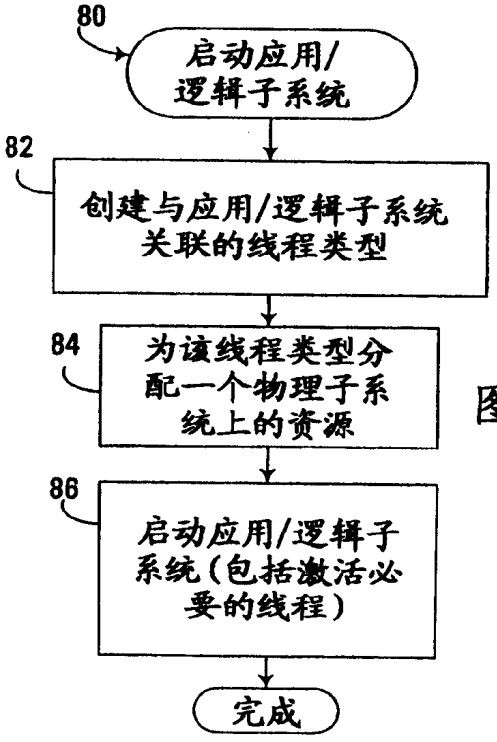

80
启动应用/逻辑子系统
82
创建与应用/逻辑子系统关联的线程类型
84
为该线程类型分配一个物理子系统上的资源
86
启动应用/逻辑子系统(包括激活必要的线程)
完成

图5

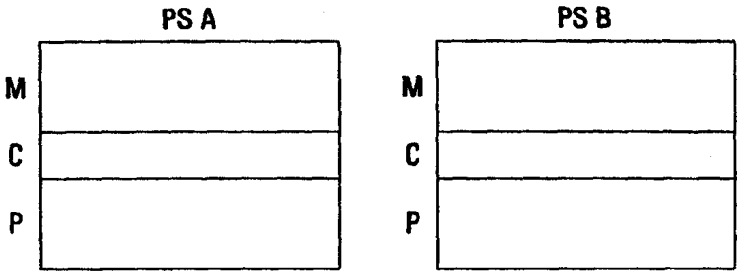


图6A

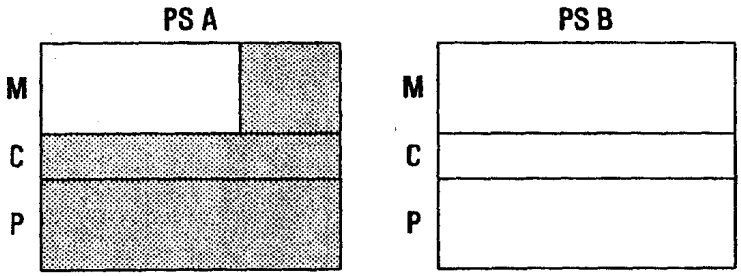


图6B

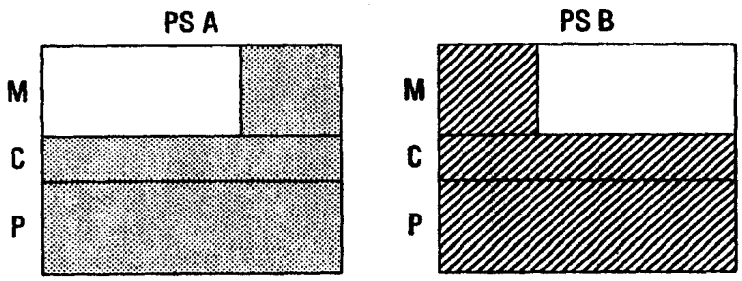


图6C

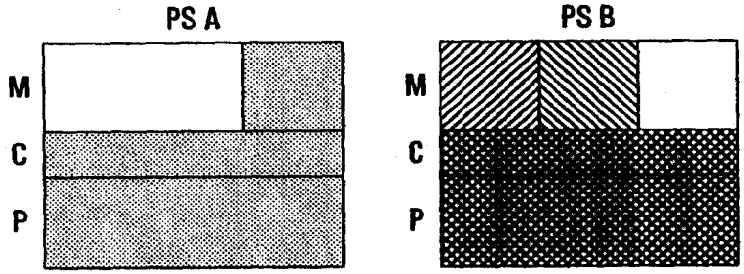


图6D

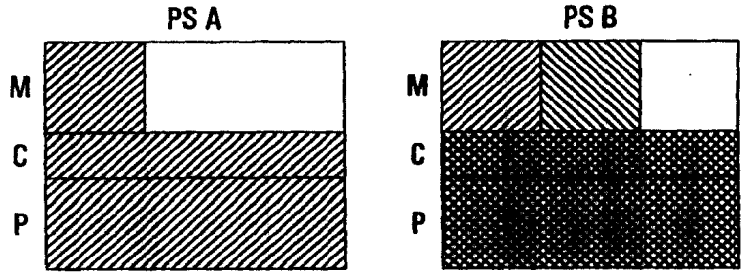


图7